L 12

```
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
```

```
**FILE**ID**BUGCHECK                                C 3
```

```
BBBBBBBB    UU      UU    GGGGGGGG    CCCCCCCC  HH      HH  EEEEEEEEEE    CCCCCCCC  KK      KK
BBBBBBBB    UU      UU    GGGGGGGG    CCCCCCCC  HH      HH  EEEEEEEEEE    CCCCCCCC  KK      KK
BB      BB  UU      UU  GG          CC          HH      HH  EE          CC          KK      KK
BB      BB  UU      UU  GG          CC          HH      HH  EE          CC          KK      KK
BB      BB  UU      UU  GG          CC          HH      HH  EE          CC          KK    KK
BB      BB  UU      UU  GG          CC          HH      HH  EE          CC          KK    KK
BBBBBBBB    UU      UU  GG          CC          HHHHHHHHHH  EEEEEEEE    CC          KKKKKK
BBBBBBBB    UU      UU  GG          CC          HHHHHHHHHH  EEEEEEEE    CC          KKKKKK
BB      BB  UU      UU  GG  GGGGGG  CC          HH      HH  EE          CC          KK    KK
BB      BB  UU      UU  GG  GGGGGG  CC          HH      HH  EE          CC          KK    KK
BB      BB  UU      UU  GG      GG  CC          HH      HH  EE          CC          KK      KK      ....
BB      BB  UU      UU  GG      GG  CC          HH      HH  EE          CC          KK      KK      ....
BBBBBBBB    UUUUUUUUUU    GGGGGG      CCCCCCCC  HH      HH  EEEEEEEEEE    CCCCCCCC  KK      KK      ....
BBBBBBBB    UUUUUUUUUU    GGGGGG      CCCCCCCC  HH      HH  EEEEEEEEEE    CCCCCCCC  KK      KK      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    III!II    SSSSSSSS
```

Table of contents

```
0000          .TITLE  BUGCHECK - SOFTWARE BUG CHECK ERROR LOGIC
0000          .IDENT  'V04-000'
0000      ;
0000      ;****************************************************************************
0000      ;*                                                                          *
0000      ;*   COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                *
0000      ;*   DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                 *
0000      ;*   ALL RIGHTS RESERVED.                                                   *
0000      ;*                                                                          *
0000      ;*   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
0000      ;*   ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH LICENSE  AND WITH THE   *
0000      ;*   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
0000      ;*   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
0000      ;*   OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
0000      ;*   TRANSFERRED.                                                           *
0000      ;*                                                                          *
0000      ;*   THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
0000      ;*   AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000      ;*   CORPORATION.                                                           *
0000      ;*                                                                          *
0000      ;*   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
0000      ;*   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
0000      ;*                                                                          *
0000      ;*                                                                          *
0000      ;****************************************************************************
0000      ;
0000      ; D. N. CUTLER 29-OCT-77
0000      ;
0000      ; SOFTWARE BUG CHECK ERROR LOGIC
0000      ;
0000      ; MODIFICATION HISTORY:
0000      ;
0000      ;       V03-011 KTA3113         Kerbey T. Altmann       21-Mar-1984
0000      ;               Add support for calling a SCS shutdown routine.
0000      ;               Put in a halt if bugcheck code cannot be read.
0000      ;               Add some more comments.
0000      ;
0000      ;       V03-010 MSH0008         Michael S. Harvey       10-Feb-1984
0000      ;               Don't display image name if no image is active.
0000      ;
0000      ;       V03-009 KDM0049         Kathleen D. Morse       08-Jul-1983
0000      ;               Move ICR, TODR, and ACCS to cpu-dependent register
0000      ;               dump routine.
0000      ;
0000      ;       V03-008 KTA3060         Kerbey T. Altmann       22-Jun-1983
0000      ;               Add code to call a possible unit disconnect routine
0000      ;               in bootdriver after shutdown.
0000      ;
0000      ;       V03-007 ROW0188         Ralph O. Weber          30-APR-1983
0000      ;               Fix truncation errors to ERL$ routines.
0000      ;
0000      ;       V03-006 TCM0003         Trudy C. Matthews       16-Feb-1983
0000      ;               Initialize console registers in a CPU-dependent fashion
0000      ;               before doing I/O to console terminal.
0000      ;
0000      ;       V03-005 TCM0002         Trudy C. Matthews       16-Dec-1982
0000      ;               Initialize R2 before calling CON$SENDCONSCMD.
```

```
                   0000  ;
                   0000  ;       V03-004 TCM0001         Trudy C. Matthews       10-Nov-1982
                   0000  ;               Call CPU-dependent routine CON$SENDCONSCMD to send "reboot
                   0000  ;               CPU" command to the console.
                   0000  ;
                   0000  ;       V03-003 ROW0120         Ralph O. Weber          24-AUG-1982
                   0000  ;               Change EXE$BOOTCB_CHK to not include the WCB$L_READS and the
                   0000  ;               WCB$L_WRITES fields of the SYS.EXE window control block in the
                   0000  ;               boot control block / SYS.EXE window control block checksum.
                   0000  ;               Paging I/O counts page reads/writes in these fields thus
                   0000  ;               causing a checksum test which includes them to fail
                   0000  ;               unnecessarily.
                   0000  ;
                   0000  ;       V03-002 KDM0002         Kathleen D. Morse       28-Jun-1982
                   0000  ;               Added $IODEF.
                   0000  ;
                   0000  ;
                   0000  ; MACRO LIBRARY CALLS
                   0000  ;
                   0000
                   0000          $BOODEF                         ;DEFINE BOOT CONTROL BLOCK OFFSETS
                   0000          $BQODEF                         ;DEFINE BOOT QIO OFFSETS
                   0000          $CONDEF                         ;DEFINE CONSOLE FUNCTION CODES
                   0000          $DMPDEF                         ;DEFINE DUMP FILE HEADER BLOCK
                   0000          $EMBDEF <CR,BC>                 ;DEFINE EMB OFFSETS
                   0000          $IFDDEF                         ;IMAGE FILE DESCRIPTOR DEFINITIONS
                   0000          $IODEF                          ;DEFINE I/O FUNCTION CODES
                   0000          $MBADEF                         ;MASS BUS ADAPTER INITIALIZATION
                   0000          $PCBDEF                         ;DEFINE PCB OFFSETS
                   0000          $PFNDEF                         ;DEFINE PFN DATA BASE BITS AND FIELDS
                   0000          $PRDEF                          ;DEFINE PROCESSOR REGISTERS
                   0000          $PRVDEF                         ;DEFINE PRIVILEGE BITS
                   0000          $PTEDEF                         ;DEFINE PAGE TABLE BITS AND FIELDS
                   0000          $PSLDEF                         ;DEFINE PROCESSOR STATUS BITS
                   0000          $RPBDEF                         ;DEFINE RESTART PARAMETER BLOCK
                   0000          $SSDEF                          ;DEFINE SYSTEM STATUS VALUES
                   0000          $STSDEF                         ;DEFINE STATUS CODE FIELDS
                   0000          $UBADEF                         ;DEFINE UNIBUS ADAPTER VALUES
                   0000          $VADEF                          ;DEFINE VIRTUAL ADDRESS FIELDS
                   0000          $WCBDEF                         ;DEFINE WINDOW CONTROL BLOCK OFFSETS
                   0000
                   0000  ;
                   0000  ; LOCAL SYMBOLS
                   0000  ;
               00000000          .PSECT  $$$025
                   0000
                   0000  BUGCHK_FLAGS:                           ;FLAGS TO BE USED BY BUGCHECK CODE
          00000000 0000          .LONG   0
                   0004  FATAL_SPSAV:
          00000000 0004          .LONG   0                       ;FATAL BUGCHECK IN PROGRESS SP
                   0008  EXE$GL_BUGCHECK::                       ;SAVED FATAL BUGCHECK CODE
          00000000 0008          .LONG   0                       ;
                   000C  ;
                   000C  ; CHARACTER CODE DEFINITIONS
                   000C  ;
                   000C
          0000000D 000C  CR=13                                   ;CARRIAGE RETURN
```

```
                     0000000A  000C   115 LF=10                                        ;LINE FEED
                               000C   116
                               000C   117 ;
                               000C   118 ; LOCAL DATA
                               000C   119 ;
                           00000000   120           .PSECT  $ZBUGFATAL,WORD            ;PSECT TO LOCATE EXECUTION LOCATION FOR
                               0000   121                                              ;BUGCHECK
                               0000   122 BUG$FATAL::                                  ;MARKER ADDRESS
                               0000   123
                           00000000   124           .PSECT  Z$INIT__BUGZEND,WORD       ;END OF BUGCHECK PSECTS
                               0000   125 BUG$A_PAGEDEND::                             ;
                               0000   126
                               0000   127 ;
                               0000   128 ; BUG CHECK MESSAGE CONTROL TEXT
                               0000   129 ;
                               0000   130
                           00000000   131           .PSECT  Z$INIT__BUGC
                               0000   132 PRCNAM_MSG:
52 52 55 43 20 20 20 20 0A 0A 0D 00'  0000   133           .ASCIC  <CR><LF><LF>/    CURRENT PROCESS = /
20 53 53 45 43 4F 52 50 20 54 4E 45   000C
                              20 3D   0018
                                 19   0000
                                      001A   134 PRCPRV_MSG:
43 4F 52 50 20 20 20 20 0A 0A 0D 00'  001A   135           .ASCIC  <CR><LF><LF>/    PROCESS PRIVILEGES/<CR><LF><LF>
47 45 4C 49 56 49 52 50 20 53 53 45   0026
                  0A 0A 0D 53 45      0032
                                 1C   001A
                                      0037   136 IMGNAM_MSG:
47 41 4D 49 20 20 20 20 0A 0A 0D 00'  0037   137           .ASCIC  <CR><LF><LF>/    IMAGE NAME = /
         20 3D 20 45 4D 41 4E 20 45   0043
                                 14   0037
                                      004C   138 SHUT_DOWN:                                 ;OPERATOR REQUESTED SHUTDOWN
48 53 20 4D 45 54 53 59 53 09 0A 0D   004C   139           .ASCII  <CR><LF>/      SYSTEM SHUTDOWN COMPLETE - /    ;
4C 50 4D 4F 43 20 4E 57 4F 44 54 55   0058
                  20 2D 20 45 54 45   0064
20 45 4C 4F 53 4E 4F 43 20 45 53 55   006A   140           .ASCIZ  /USE CONSOLE TO HALT SYSTEM/<CR><LF>
54 53 59 53 20 54 4C 41 48 20 4F 54   0076
                  00 0A 0D 4D 45      0082
                                      0087   141 MSGCTRL:                                   ;
41 54 41 46 20 2A 2A 2A 2A 0A 0A 0D   0087   142           .ASCIZ  <CR><LF><LF>/**** FATAL BUG CHECK, VERSION = / ;
2C 4B 43 45 48 43 20 47 55 42 20 4C   0093
00 20 3D 20 4E 4F 49 53 52 45 56 20   009F
                                      00AB   143 MSGCTRL1:
45 54 53 49 47 45 52 20 20 20 20 0A   00AB   144           .ASCII  <LF>/    REGISTER DUMP/<CR><LF><LF> ;
      0A 0A 0D 50 4D 55 44 20 52      00B7
         00 20 3D 20 30 52 09         00C0   145           .ASCIZ  /        R0 = /            ;
         00 20 3D 20 31 52 09         00C7   146           .ASCIZ  /        R1 = /            ;
         00 20 3D 20 32 52 09         00CE   147           .ASCIZ  /        R2 = /            ;
         00 20 3D 20 33 52 09         00D5   148           .ASCIZ  /        R3 = /            ;
         00 20 3D 20 34 52 09         00DC   149           .ASCIZ  /        R4 = /            ;
         00 20 3D 20 35 52 09         00E3   150           .ASCIZ  /        R5 = /            ;
         00 20 3D 20 36 52 09         00EA   151           .ASCIZ  /        R6 = /            ;
         00 20 3D 20 37 52 09         00F1   152           .ASCIZ  /        R7 = /            ;
         00 20 3D 20 38 52 09         00F8   153           .ASCIZ  /        R8 = /            ;
         00 20 3D 20 39 52 09         00FF   154           .ASCIZ  /        R9 = /            ;
         00 20 3D 30 31 52 09         0106   155           .ASCIZ  /        R10= /            ;
         00 20 3D 31 31 52 09         010D   156           .ASCIZ  /        R11= /            ;
```

```
               00 20 3D 20 50 41 09  0114        .ASCIZ  /       AP = /              ;
               00 20 3D 20 50 46 09  011B        .ASCIZ  /       FP = /              ;
               00 20 3D 20 50 53 09  0122        .ASCIZ  /       SP = /              ;
               00 20 3D 20 43 50 09  0129        .ASCIZ  /       PC = /              ;
               00 20 3D 4C 53 50 09  0130        .ASCIZ  /       PSL= /              ;
2F 4C 45 4E 52 45 4B 20 20 20 20 0A  0137        .ASCII  <LF>^     KERNEL/INTERRUPT STACK^<CR><LF><LF><128> ;
54 53 20 54 50 55 52 52 45 54 4E 49  0143
               80 0A 0A 0D 4B 43 41  014F
54 53 20 43 45 58 45 20 20 20 20 0A  0156        .ASCII  <LF>/     EXEC STACK/<CR><LF><LF><128> ;
               80 0A 0A 0D 4B 43 41  0162
                                     0169
                                     0169   ;
                                     0169   ; PROCESSOR REGISTER DUMP CONTROL TABLE
                                     0169   ;
                                     0169
                                 00000000        .PSECT  $AEXENONPAGED
                                     0000   REGTAB:                                 ;
                               00    0000        .BYTE   PR$_KSP                    ;KERNEL STACK POINTER
                               01    0001        .BYTE   PR$_ESP                    ;EXECUTIVE STACK POINTER
                               02    0002        .BYTE   PR$_SSP                    ;SUPERVISOR STACK POINTER
                               03    0003        .BYTE   PR$_USP                    ;USER STACK POINTER
                               04    0004        .BYTE   PR$_ISP                    ;INTERRUPT STACK POINTER
                               80    0005        .BYTE   128                        ;TABLE ESCAPE
                               08    0006        .BYTE   PR$_P0BR                   ;PROGRAM REGION BASE REGISTER
                               09    0007        .BYTE   PR$_P0LR                   ;PROGRAM REGION LIMIT REGISTER
                               0A    0008        .BYTE   PR$_P1BR                   ;CONTROL REGION BASE REGISTER
                               0B    0009        .BYTE   PR$_P1LR                   ;CONTROL REGION LIMIT REGISTER
                               0C    000A        .BYTE   PR$_SBR                    ;SYSTEM BASE REGISTER
                               0D    000B        .BYTE   PR$_SLR                    ;SYSTEM LIMIT REGISTER
                               10    000C        .BYTE   PR$_PCBB                   ;PROCESS CONTROL BLOCK BASE REGISTER
                               11    000D        .BYTE   PR$_SCBB                   ;SYSTEM CONTROL BLOCK BASE REGISTER
                               13    000E        .BYTE   PR$_ASTLVL                 ;AST DELIVERY LEVEL REGISTER
                               15    000F        .BYTE   PR$_SISR                   ;SOFTWARE INTERRUPT SUMMARY REGISTER
                               18    0010        .BYTE   PR$_ICCS                   ;INTERVAL TIMER CONTROL REGISTER
                               80    0011        .BYTE   128                        ;TABLE ESCAPE
```

```
                       0012         .SBTTL  BUG CHECK ERROR MESSAGE PROCESSING
                       0012 ;+
                       0012 ; EXE$BUG_CHECK - BUG CHECK ERROR MESSAGE PROCESSING
                       0012 ;
                       0012 ;
                       0012 ; THIS ROUTINE IS CALLED BY EXECUTING THE OPERATION CODES ^XFEFF AND
                       0012 ; X^FDFF, WHICH ARE RESERVED FOR DIGITAL AND ARE GUARANTEED TO ALWAYS
                       0012 ; CAUSE AN EXCEPTION.
                       0012 ;
                       0012 ; THIS ROUTINE CONTAINS A HOOK FOR LOADABLE MULTI-PROCESSING CODE.
                       0012 ; THE HOOK, MPH$BUGCHKHK, MUST BE LOCATED ON THE "JSB EXE$ADPINIT"
                       0012 ; INSTRUCTION.  AFTER EXECUTING SOME MULTI-PROCESSING SPECIFIC CODE,
                       0012 ; EXECUTION WILL BE CONTINUED BY JUMPING TO EXE$ADPINIT AND THEN
                       0012 ; RETURNING TO THE IN-LINE CODE IN THIS ROUTINE.
                       0012 ;
                       0012 ; INPUTS:
                       0012 ;
                       0012 ;       THE CURRENT PROCESS PCB.
                       0012 ;       THE ENTIRE PROCESSOR STATE (I.E. GENERAL REGISTERS, ETC.).
                       0012 ;       THE BUG CHECK CODE WHICH FOLLOWS IMMEDIATELY INLINE.
                       0012 ;
                       0012 ; OUTPUTS:
                       0012 ;
                       0012 ;       IF THE PREVIOUS MODE WAS KERNEL OR EXECUTIVE AND THE BUG SEVERITY IS
                       0012 ;       GREATER THAN OR EQUAL TO ERROR, THEN THE SYSTEM IS SHUT DOWN IN AN
                       0012 ;       ORDERLY FASHION BY EXECUTING THE CRASH RESTART ROUTINE.  THE CODE
                       0012 ;       TO HANDLE A FATAL BUGCHECK IS READ FROM THE SYSTEM IMAGE OVER SOME
                       0012 ;       OF THE PURE EXEC CODE USING THE SAVED BOOTSTRAP DRIVER.
                       0012 ;
                       0012 ;       IF THE PREVIOUS MODE WAS KERNEL OR EXECUTIVE AND THE BUG SEVERITY IS
                       0012 ;       LESS THAN ERROR, THEN AN ERROR LOG ENTRY IS MADE AND EXECUTION OF THE
                       0012 ;       SYSTEM CONTINUES.
                       0012 ;
                       0012 ;       IF THE PREVIOUS MODE WAS SUPERVISOR OR USER AND THE PROCESS HAS THE
                       0012 ;       PRIVILEGE TO CAUSE BUG CHECK ERROR LOG ENTRIES, THEN AN ENTRY IS MADE
                       0012 ;       IN THE ERROR LOG. OTHERWISE NO ENTRY IS MADE.
                       0012 ;
                       0012 ;       IF THE PREVIOUS MODE WAS SUPERVISOR OR USER AND THE BUG SEVERITY IS
                       0012 ;       GREATER THAN OR EQUAL TO ERROR, THEN AN EXIT SYSTEM SERVICE IS PERFORMED
                       0012 ;       ON BEHALF OF THE PROCESS AT THE MODE CAUSING THE BUG CHECK. IF THE BUG
                       0012 ;       SEVERITY IS LESS THAN ERROR, THEN EXECUTION OF THE PROCESS IS RESUMED.
                       0012 ;
                       0012 ;       IF AN ACCESS VIOLATION IS DETECTED WHILE ATTEMPTING TO FETCH THE BUG
                       0012 ;       CHECK CODE, THE EXCEPTION IS TURNED INTO AN ACCESS VIOLATION.
                       0012 ;-
                       0012
                       0012 EXE$BUG_CHECK::                             ;BUG CHECK ERROR PROCESSING
                       0012         .ENABL LSB
       7FFF 8F   BB    0012         PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP,FP,SP> ;SAVE
  50     3C AE   D0    0016         MOVL    15*4(SP),R0                 ;GET ADDRESS OF INSTRUCTION
                       001A         IFRD    #2,2(R0),20$                ;CAN LOWER HALF OF BUG CHECK CODE BE READ?
       7FFF 8F   BA    0021 10$:    POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP,FP,SP> ;RESTORE
            6E   DD    0025         PUSHL   (SP)                        ;DUPLICATE ADDRESS OF INSTRUCTION
       6E   02   C0    0027         ADDL    #2,(SP)                     ;CALCULATE ADDRESS OF VIOLATION
            00   DD    002A         PUSHL   #0                          ;SET REASON CODE
         FFD1'   31    002C         BRW     EXE$ACVIOLAT                ;
                       002F
```

```
                    002F   ;
                    002F   ; BUG CHECK CODE CAN BE READ
                    002F   ;
                    002F
   5D    02 A0   3C 002F   20$:    MOVZWL   2(R0),FP                         ;GET LOWER HALF OF BUGCHECK CODE
   3C AE    04   C0 0033           ADDL     #4,15*4(SP)                      ;CALCULATE ADDRESS OF NEXT INSTRUCTION
      5C    5E   D0 0037           MOVL     SP,AP                            ;SET ADDRESS OF SAVED REGISTERS
   5B    C3 AF   9E 003A           MOVAB    REGTAB,R11                       ;GET ADDRESS OF PROCESSOR REGISTER TABLE
            5A   DC 003E           MOVPSL   R10                              ;READ CURRENT PROCESSOR STATUS
 01 A0   FD 8F   91 0040           CMPB     #^XFD,1(R0)                      ;BUG CHECK LONG?
            0F   12 0045           BNEQ     25$                              ;IF NEQ NO
                    0047           IFNORD   #2,4(R0),10$                     ;CAN UPPER HALF OF BUG CHECK CODE BE READ?
   5D    02 A0   D0 004E           MOVL     2(R0),FP                         ;GET BUG CHECK CODE
   3C AE    02   C0 0052           ADDL     #2,15*4(SP)                      ;CALCULATE ADDRESS OF NEXT INSTRUCTION
      02    16   ED 0056   25$:    CMPZV    #PSL$V_PRVMOD,#PSL$S_PRVMOD,- ;PREVIOUS MODE EXEC OR KERNEL?
      01    5A      0059           R10,#PSL$C_EXEC                  ;
            64   15 005B           BLEQ     70$                              ;IF LEQ YES
  54    0000'CF  D0 005D           MOVL     W^SCH$GL_CURPCB,R4               ;GET CURRENT PROCESS PCB ADDRESS
                    0062           IFNPRIV  BUGCHK,40$                       ;DOES PROCESS HAVE PRIVILEGE TO BUG CHECK?
  59    0070 8F  3C 0068           MOVZWL   #EMB$K_UBC,R9                    ;SET ENTRY TYPE
  51    0080 8F  3C 006D   30$:    MOVZWL   #EMB$K_BC_LENGTH,R1              ;GET LENGTH OF BUGCHECK MESSAGE
     00000000'EF 16 0072           JSB      ERL$ALLOCEMB                     ;ALLOCATE BUG CHECK ERROR MESSAGE BUFFER
         23 50   E9 0078           BLBC     R0,40$                           ;IF LBC ALLOCATION FAILURE
          00FA   30 007B           BSBW     BUILD_HEADER                     ;BUILD MESSAGE HEADER AND DUMP REGISTERS
   68 A2    5D   D0 007E           MOVL     FP,EMB$L_BC_CODE(R2)             ;SET BUGCHECK CODE INTO MESSAGE
51  00000000'9F  D0 0082           MOVL     @#SCH$GL_CURPCB,R1               ;GET ADR OF CURRENT PROCESS'S PCB
         60 A1   D0 0089           MOVL     PCB$L_PID(R1),-
         6C A2      008C                    EMB$L_BC_PID(R2)                 ;SET PROCESS ID INTO MESSAGE
         70 A1   7D 008E           MOVQ     PCB$T_LNAME(R1),-
         70 A2      0091                    EMB$T_BC_LNAME(R2)               ;SET PROCESS NAME INTO
         78 A1   7D 0093           MOVQ     PCB$T_LNAME+8(R1),-
         78 A2      0096                    EMB$T_BC_LNAME+8(R2)             ; ERROR LOG MESSAGE
     00000000'EF 16 0098           JSB      ERL$RELEASEMB                    ;RELEASE ERROR MESSAGE BUFFER
         0C 5D   E8 009E   40$:    BLBS     FP,50$                           ;IF LBS NONFATAL BUG CHECK
      03    00   ED 00A1           CMPZV    #STS$V_SEVERITY,#STS$S_SEVERITY,- ;FATAL BUG CHECK?
      02    5D      00A4                    FP,#STS$K_ERROR                  ;
            05   19 00A6           BLSS     50$                              ;IF LSS NO
 3C AE   B2'AF   9E 00A8           MOVAB    B^60$,15*4(SP)                   ;REPLACE RETURN PC
       7FFF 8F   BA 00AD   50$:    POPR     #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP,FP,SP> ;RESTORE
                 02 00B1           REI                                       ;
                    00B2
                    00B2   ;
                    00B2   ; EXECUTE EXIT SYSTEM SERVICE ON BEHALF OF PROCESS
                    00B2   ;
                    00B2
                    00B2   60$:    $EXIT_S #SS$_BUGCHECK                     ;EXIT MODE
            F1   11 00BF           BRB      60$                              ;
                    00C1
                    00C1   ;
                    00C1   ; PREVIOUS MODE WAS EXECUTIVE OR KERNEL
                    00C1   ;
                    00C1
      59    28   3C 00C1   70$:    MOVZWL   #EMB$K_SBC,R9                    ;SET ENTRY TYPE
0A 0000'CF  00'  E0 00C4           BBS      S^#EXE$V_FATAL_BUG,W^EXE$GL_FLAGS,75$ ;IF SET, ALL FATAL
         A0 5D   E8 00CA           BLBS     FP,30$                           ;IF LBS NONFATAL BUG CHECK
      03    00   ED 00CD           CMPZV    #STS$V_SEVERITY,#STS$S_SEVERITY,- ;FATAL BUG CHECK?
      02    5D      00D0                    FP,#STS$K_ERROR                  ;
```

```
                  99   19  00D2             BLSS    30$                         ;IF LSS NO
                           00D4
                00C0   30  00D4  75$:       BSBW    EXE$BOOTCB_CHK              ;IS BOOT CONTROL BLOCK OK?
                  03   13  00D7             BEQL    80$                         ;BRANCH IF YES
               24 A1   D4  00D9             CLRL    BOO$L_BUG_MAP(R1)           ;NO, SET UP TO ISSUE REBOOT
                           00DC
                           00DC  ;
                           00DC  ; SHUT DOWN SYSTEM IN AN ORDERLY MANNER
                           00DC  ;
                           00DC
                           00DC  80$:       SETIPL  #31                         ;DISABLE ALL INTERRUPTS
      06 0000'CF     00    E2  00DF         BBSS    #0,W^BUGCHK_FLAGS,82$       ;ONLY DO THIS ONCE
         00000000'GF       16  00E5         JSB     G^SCS$SHUTDOWN              ;CALL SCS DO SHUTDOWN ALL CIRCUITS
       56    0000'CF       D0  00EB  82$:   MOVL    W^EXE$GL_RPB,R6             ;GET ADDRESS OF RESTART PARAMETERS
         00000000'GF       16  00F0         JSB     G^EXE$SHUTDWNADP            ;SHUT DOWN ANY ADAPTERS THAT NEED IT
                           00F6  MPH$BUGCHKHK::                                 ;MULTI-PROCESSING HOOK (REPLACES JSB)
         00000000'GF       16  00F6         JSB     G^EXE$INIBOOTADP            ;INIT BOOT DEVICE ADAPTER BEFORE
                           00FC                                                 ; READING FATAL BUGCHECK CODE
                           00FC                                                 ;DO NOT STEP OVER NEXT 2 LINES OR PAGES
                           00FC                                                 ;WILL BE SET RDONLY AFTER JSB BY XDELTA
         00000000'EF       16  00FC         JSB     INI$WRITABLE                ;MAKE SYSTEM CODE WRITEABLE
      00 0000'CF     00'   E2  0102         BBSS    S^#EXE$V_SYSWRTABL,W^EXE$GL_FLAGS,85$ ; INHIBIT INI$RDONLY/WRITABLE
                           0108                                                 ; (RDONLY CALLED ON EVERY XDELTA EXIT)
            53    34 A6    D0  0108  85$:   MOVL    RPB$L_IOVEC(R6),R3          ;FETCH POINTER TO BOOTDRIVER
                           010C  ;
                           010C  ; CHECK THE VMB VERSION NUMBER.  IF IT EXISTS AND IF IT IS 7 OR GREATER, THEN
                           010C  ; CALL A UNIT INITIALIZATION ROUTINE TO DO ANY DEVICE/UNIT SPECIFIC INIT
                           010C  ; THAT IS NOT DONE IN ADAPTER INIT.
                           010C  ;
            50    10 A3    B2  010C         MCOMW   BQO$W_VERSION(R3),R0        ;GET VMB VERSION NUMBER 1'S COMPLEMENTED
            12 A3    50    B1  0110         CMPW    R0,BQO$W_VERSION+2(R3)      ;CHECK AGAINST CHECK WORD IN VMB
                     16    12  0114         BNEQ    90$                         ;IF NOT, ASSUME NO VERSION NUMBER
            07    10 A3    B1  0116         CMPW    BQO$W_VERSION(R3),#7        ;VERSION 7 OR GREATER OF VMB?
                     10    1F  011A         BLSSU   90$                         ;NO, DON'T CALL THE NON-EXISTENT CODE
            55    1C A3    D0  011C         MOVL    BQO$L_UNIT_INIT(R3),R5      ;YES, IS THE ROUTINE PRESENT?
                     0A    13  0120         BEQL    90$                         ;NO, DON'T CALL
                  59 56    D0  0122         MOVL    R6,R9                       ;YES, SHIFT INPUT PARAMETERS
              6345   00    FB  0125         CALLS   #0,(R3)[R5]                 ;DO IT!
                  32 50    E9  0129         BLBC    R0,REBOOT                   ;INIT FAILED, JUST REBOOT
   55    00000000'GF       D0  012C  90$:   MOVL    G^EXE$GL_BOOTCB,R5          ;ADDRESS OF BOOT CONTROL BLOCK
            52    24 A5    D0  0133         MOVL    BOO$L_BUG_MAP(R5),R2        ;VIRTUAL TO LOGICAL MAP FOR
                           0137                                                 ;NON-RESIDENT BUGCHECK CODE AND DATA
                     25    13  0137         BEQL    REBOOT                      ;REBOOT IF BAD BOOT CONTROL BLOCK
                     56    DD  0139         PUSHL   R6                          ;SET RPB ADDRESS IN ARGUMENT LIST
                     01    DD  013B         PUSHL   #1                          ;SET FOR VIRTUAL ADDRESS I/O
                     21    DD  013D         PUSHL   S^#IO$_READLBLK             ;SET FUNCTION TO READ
                  08 A2    DD  013F         PUSHL   8(R2)                       ;STARTING LBN
      7E    04 A2    09    78  0142         ASHL    #9,4(R2),-(SP)              ;NO. OF BYTES IN THIS PIECE
              0000'CF      9F  0147         PUSHAB  W^BUG$FATAL                 ;BUFFER ADDRESS
                     06    DD  014B         PUSHL   #6                          ;NO. OF ARGUMENTS
      00 B343        6E    FA  014D         CALLG   (SP),@BQO$L_QIO(R3)[R3]     ;CALL BOOTDRIVR TO READ FOLLOWING CODE
               03    50    E9  0152         BLBC    R0,READ_ERR_RETRY           ;BR IF ERROR TO RETRY
                  04 BE    17  0155         JMP     @4(SP)                      ;JUMP TO FATALBUG
                           0158
                           0158  READ_ERR_RETRY:
            5E    1C       C0  0158         ADDL    #7*4,SP                     ;CLEAN OFF ARG LIST
                FF7E       31  015B         BRW     80$                         ;TRY READ AGAIN
```

```
                          015E         .DSABL LSB
                          015E  ;
                          015E  ; REBOOT THE PROCESSOR
                          015E  ;
                          015E  ; Control comes here on 4 paths:
                          015E  ;         1) Boot control block failed its checksum-verification
                          015E  ;         2) Failure to initialize boot device/path
                          015E  ;         3) Failure attempting to read first block of bugcheck code
                          015E  ;         4) Dump terminated successfully and BUGREBOOT was on
                          015E  ;
                          015E  REBOOT:
                00'   E1  015E         BBC     S^#EXE$V_BUGREBOOT,-
         0B 00000000'9F   0160                 @#EXE$GL_FLAGS,10$        ;BRANCH IF NO REBOOT
              50   02 D0  0166         MOVL    #CON$C_BOOTCPU,R0         ;CONSOLE FUNCTION = REBOOT
                   52 D4  0169         CLRL    R2                        ;SIGNAL NO RETURN DATA EXPECTED
         00000000'EF  16  016B         JSB     CON$SENDCONSCMD           ;CALL CPU-DEPENDENT ROUTINE
                          0171  ;
                          0171  ; CONTROL NEVER RETURNS HERE.
                          0171  ;
         00000000'9F  16  0171  10$:   JSB     @#INI$BRK                 ;STOP IN XDELTA IF PRESENT
                      00  0177         HALT
```

```
                               0178         .SBTTL  NON-RESIDENT BUG CHECK CODE
                               0178
                               0178  ;
                               0178  ; READ IN THE REST OF THE BUGCHECK CODE AND DATA THAT WAS NOT CONTIGUOUS
                               0178  ; WITH THIS FIRST PART.  THE FOLLOWING CODE MUST BE TOTALLY CONTAINED
                               0178  ; IN THE FIRST PAGE OF THE NON-RESIDENT BUGCHECK CODE TO BE CERTAIN
                               0178  ; THAT IT IS READ BY THE FIRST READ IN THE RESIDENT PORTION.
                               0178  ;
                               0178  ; THE FOLLOWING STATE IS ASSUMED:
                               0178  ;       R2 = VIRTUAL TO LOGICAL MAP FOR NON-RESIDENT BUGCHECK CODE AND DATA
                               0178  ;       R3 = RPB$L_IOVEC(RPB)
                               0178  ;       THE FIRST SEVEN LONG WORDS ON THE STACK ARE THE ARGUMENT LIST
                               0178  ;               TO BOO$QIO IN THE BOOT DRIVER.
                               0178  ;
                               0178
                           00000000         .PSECT  Z$INIT__BUGA,PAGE           ;FIRST BUGCHECK PSECT IN INIT REGION
                               0000
                               0000  BUG$A_PAGED::
                               0000  FATALBUG:                                  ;START OF FATAL BUGCHECK CODE
 57    82    FD 8F       78    0000         ASHL    #-3,(R2)+,R7               ;GET COUNT OF RETRIEVAL POINTERS
          52    08       C0    0005         ADDL    #8,R2                      ;POINT TO SECOND RETRIEVAL POINTER
                1C       11    0008         BRB     30$                        ;ALREADY DONE FIRST POINTER
   04 AE    08 AE        C0    000A  20$:   ADDL    8(SP),4(SP)                ;ADJUST XFER ADR BY BYTE COUNT READ
 08 AE    82    09       78    000F         ASHL    #9,(R2)+,8(SP)             ;BYTE COUNT FOR NEXT PIECE
      0C AE     82       D0    0014         MOVL    (R2)+,12(SP)               ;LBN FOR NEXT PIECE
   00 B343      6E       FA    0018         CALLG   (SP),@BOO$L_QIO(R3)[R3]    ;READ BUGCHECK CODE AND DATA
          06    50       E8    001D         BLBS    R0,30$                     ;BRANCH IF OK
     00000158'9F         17    0020         JMP     @#READ_ERR_RETRY           ;ERROR - TRY THE WHOLE THING OVER
          E1    57       F5    0026  30$:   SOBGTR  R7,20$                     ;READ EVERYTHING IN THE MAP
          5E    1C       C0    0029         ADDL    #7*4,SP                    ;CLEAN OFF THE ARG LIST
                               002C  ;
                               002C  ; END OF CODE THAT MUST BE TOTALLY CONTAINED IN THE FIRST PAGE OF
                               002C  ; NON-RESIDENT BUGCHECK CODE.
                               002C  ;
 50    00000004'9F       9E    002C         MOVAB   @#FATAL_SPSAV,R0           ;ADDRESS OF SAVED FATAL SP
                60       D5    0033         TSTL    (R0)                       ;ALREADY IN A FATAL BUGCHECK?
                06       13    0035         BEQL    82$                        ;BRANCH IF NOT
          5E    60       D0    0037         MOVL    (R0),SP                    ;RESTORE SP FROM PREVIOUS BUGCHECK
              01AA       31    003A         BRW     CONSOLE_DONE               ;AND GO REBOOT THE SYSTEM
          60    5E       D0    003D  82$:   MOVL    SP,(R0)                    ;NOTE THAT WE ARE IN A FATAL BUGCHECK
    00 5A       16       E2    0040         BBSS    #PSL$V_PRVMOD,R10,84$      ;JAM PREVIOUS MODE TO EXEC
                               0044                                            ;THUS FORCING EXEC STACK DUMP TOO
                               0044  ;
                               0044  ; NOW BUILD THE DUMP FILE HEADER BLOCK.  A PIECE OF SYSTEM SPACE IS
                               0044  ; USED FOR THE BUFFER SINCE THIS IS THE ONLY ADDRESSES FOR WHICH I/O
                               0044  ; CAN BE DONE.  THE CRASH ERROR LOG ENTRY IS BUILT IN THIS BUFFER TO
                               0044  ; GUARANTEE THAT IS IS INCLUDED IN THE DUMP, (SINCE THE ERROR LOG BUFFERS
                               0044  ; MAY BE FULL).
                               0044  ;
        FE28 CF          9E    0044  84$:   MOVAB   FATALBUG-512+DMP$C_LENGTH+EMB$K_LENGTH,- ;BUFFER ADDRESS IS
                52             0048                 R2                         ;THE PAGE PREVIOUS TO THIS CODE
     00F4 C2    5D       D0    0049         MOVL    FP,EMB$L_CR_CODE(R2)       ;SET BUGCHECK CODE INTO MESSAGE
 51    00000000'9F       D0    004E         MOVL    @#SCH$GL_CURPCB,R1         ;GET ADR OF CURRENT PROCESS'S PCB
 00F8 C2    60 A1        D0    0055         MOVL    PCB$L_PID(R1),EMB$L_CR_PID(R2) ;SET PROCESS ID INTO MESSAGE
 00FC C2    70 A1        7D    005B         MOVQ    PCB$T_LNAME(R1),EMB$T_CR_LNAME(R2) ;SET PROCESS NAME INTO
 0104 C2    78 A1        7D    0061         MOVQ    PCB$T_LNAME+8(R1),EMB$T_CR_LNAME+8(R2) ; ERROR LOG MESSAGE
 FC A2    010C 8F        3C    0067         MOVZWL  #EMB$K_CR_LENGTH,EMB$W_SIZE(R2) ;SET THE SIZE OF THIS MSG
```

```
         62    3E  DB  006D           MFPR    #PR$_SID,EMB$L_HD_SID(R2) ;SET SYSTEM ID IN MESSAGE
 06 A2  00000000'9F  7D  0070           MOVQ    @#EXE$GQ_SYSTIME,EMB$Q_CR_TIME(R2) ;SET TIME ERROR OCCURRED
 0E A2  00000000'9F  B0  0078           MOVW    @#ERL$GL_SEQUENCE,EMB$W_CR_ERRSEQ(R2) ;SET ERROR SEQUENCE NUMBER
        00000000'9F  D6  0080           INCL    @#ERL$GL_SEQUENCE       ;INCREMENT ERROR SEQUENCE NUMBER
         59    25  3C  0086           MOVZWL  #EMB$K_CR,R9            ;SET TYPE OF ENTRY
        00000178'9F  16  0089           JSB     @#BUILD_HEADER          ;BUILD HEADER AND DUMP REGISTERS
        0000018C'9F  16  008F           JSB     @#DUMP_REGISTERS        ;DUMP REMAINDER OF CPU-INDEPENDENT
                         0095                                           ; PROCESSOR REGISTERS
        00000000'9F  16  0095           JSB     @#EXE$DUMPCPUREG        ;DUMP CPU-DEPENDENT PROCESSOR
                         009B                                           ; REGISTERS
            FF A2  96  009B           INCB    EMB$B_VALID(R2)         ;INDICATE ERL ENTRY IS COMPLETE
   00000008'9F  5D  D0  009E           MOVL    FP,@#EXE$GL_BUGCHECK    ;SAVE BUGCHECK CODE
        00000000'9F  16  00A5           JSB     @#CON$OWNCTY            ;SET UP CONSOLE TERMINAL REGISTERS
   00000004'8F  5D  D1  00AB           CMPL    FP,#<BUG$_OPERATOR!STS$K_SEVERE> ;IS THIS AN OPERATOR SHUTDOWN?
               03  12  00B2           BNEQ    100$                    ;NO, CONTINUE
             0130  31  00B4           BRW     CONSOLE_DONE            ;YES, DONT GIVE NORMAL BUGCHECK MESSAGE
                         00B7   ;
                         00B7   ; OUTPUT THE BUGCHECK MESSAGE, REGISTER, AND STACK DUMP ON CONSOLE.
                         00B7   ;
               5D  DD  00B7 100$:  PUSHL   FP                      ;SAVE BUG CHECK CODE
         5C    5E  D0  00B9           MOVL    SP,AP                   ;SET ADDRESS OF REGISTERS
               5B  D4  00BC           CLRL    R11                     ;SET FOR CONSOLE TERMINAL OUTPUT
 59     00000087'EF  9E  00BE           MOVAB   MSGCTRL,R9              ;GET ADDRESS OF CONTROL TEXT
         50    89  98  00C5 110$:  CVTBL   (R9)+,R0                ;GET NEXT BYTE FROM CONTROL TEXT
               6B  19  00C8           BLSS    130$                    ;IF LSS END OF TEXT
               05  13  00CA           BEQL    120$                    ;IF EQL ESCAPE CHARACTER
            FF31'  30  00CC           BSBW    EXE$OUTCHAR             ;OUTPUT CHARACTER
               F4  11  00CF           BRB     110$                    ;
         51    8C  D0  00D1 120$:  MOVL    (AP)+,R1                ;GET NEXT LONGWORD TO CONVERT
     50    00AB'CF  9E  00D4           MOVAB   W^MSGCTRL1,R0           ;GET ADDRESS OF REGISTER STRING
         59    50  D1  00D9           CMPL    R0,R9                   ;CHECK FOR END OF HEADER
               4F  12  00DC           BNEQ    124$                    ;BRANCH IF NOT AT END
         51    7E  7E  00DE           MOVAQ   -(SP),R1                ;CREATE BUFFER FOR VERSION TEXT
 01 AE  00000000'9F  D0  00E1           MOVL    @#SYS$GQ_VERSION,1(SP)  ;SET VERSION NUMBER IN BUFFER
         6E    05  90  00E9           MOVB    #5,(SP)                 ;SET COUNT FOR VERSION AND SPACE
      05 AE    20  90  00EC           MOVB    #32,5(SP)               ;SET TRAILING SPACE
            FF0D'  30  00F0           BSBW    EXE$OUTCSTRING          ;PRINT VERSION NUMBER
         5E    08  C0  00F3           ADDL    #8,SP                   ;CLEAN STACK
     50  5D    08  C7  00F6           DIVL3   #8,FP,R0                ;CONVERT CODE TO INDEX
 51     00000000'EF  9E  00FA           MOVAB   BUG$T_MESSAGES,R1       ;SET BASE OF MESSAGES
         52    81  9A  0101 122$:  MOVZBL  (R1)+,R2                ;GET LENGTH OF MESSAGE
         51    52  C0  0104           ADDL    R2,R1                   ;AND POINT TO NEXT MESSAGE
            F7 50  F5  0107           SOBGTR  R0,122$                 ;BRANCH IF MESSAGE NOT LOCATED
            FEF3'  30  010A           BSBW    EXE$OUTCSTRING          ;OUTPUT STRING
 51     00000000'EF  DE  010D           MOVAL   PRCNAM_MSG,R1           ;"CURRENT PROCESS = "
            FEE9'  30  0114           BSBW    EXE$OUTCSTRING          ;OUTPUT COUNTED STRING
 51     00000000'9F  D0  0117           MOVL    @#SCH$GL_CURPCB,R1      ;PROCESS PCB OF CURRENT PROCESS
 51     00000070 8F  C0  011E           ADDL    #PCB$T_LNAME,R1         ;POINT AT PROCESS NAME
            FED8'  30  0125           BSBW    EXE$OUTCSTRING          ;OUTPUT PROCESS NAME COUNTED STRING
            FED5'  30  0128           BSBW    EXE$OUTCRLF             ;NEW LINE
               98  11  012B           BRB     110$                    ;
                         012D 124$:                                   ;
            FED0'  30  012D           BSBW    EXE$OUTHEX              ;OUTPUT CONVERTED HEX LONGWORD
            FECD'  30  0130           BSBW    EXE$OUTCRLF             ;OUTPUT CARRIAGE RETURN, LINE FEED PAIR
               90  11  0133 126$:  BRB     110$                    ;
      58    40 8F  9A  0135 130$:  MOVZBL  #64,R8                  ;SET LOOP COUNT
 50     00000000'9F  9E  0139           MOVAB   @#CTL$AL_STACK,R0       ;POINTER TO POSSIBLE PROCESS SPACE STACKS
```

```
51     00000000'9F   9E  0140               MOVAB   @#CTL$AL_STACKLIM,R1        ;POINTER TO POSSIBLE PROCESS STACK LIMIT
          12 5C  1F   E1  0147               BBC     #31,AP,135$                 ;BRANCH IF STACK IS IN PROCESS SPACE
50     00000000'9F   9E  014B               MOVAB   @#EXE$AL_STACKS,R0          ;POINTER TO POSSIBLE SYSTEM SPACE STACKS
     51      FC A0   DE  0152               MOVAL   -4(R0),R1                   ;USE SAME ARRAY AS LIMIT
          80     5C   D1  0156               CMPL    AP,(R0)+                    ;ADDRESS IN FIRST(NULL) STACK?
                 17   1B  0159               BLEQU   140$                        ;YES, OKAY
                 0A   11  015B               BRB     137$                        ;NO, CHECK FURTHER
                          015D      ;
                          015D      ; CHECK PROCESS KERNEL/EXEC STACKS
                          015D      ;
          80     5C   D1  015D      135$:    CMPL    AP,(R0)+                    ;ADDRESS IN FIRST STACK?
                 05   1A  0160               BGTRU   137$                        ;NO, TOO HIGH - TRY SECOND(EXEC)
          61     5C   D1  0162               CMPL    AP,(R1)                     ;BELOW FIRST STACK LIMIT?
                 0B   1A  0165               BGTRU   140$                        ;NO, ALL OKAY
          80     5C   D1  0167      137$:    CMPL    AP,(R0)+                    ;IN SECOND STACK?
                 1A   1A  016A               BGTRU   155$                        ;BRANCH IF NOT, BAD STACK ADDRESS
       04 A1     5C   D1  016C               CMPL    AP,4(R1)                    ;NOW CHECK LIMIT
                 14   1B  0170               BLEQU   155$                        ;NO, BAD STACK
    50    70     5C   C3  0172      140$:    SUBL3   AP,-(R0),R0                 ;NUMBER OF BYTES TO TOP OF STACK
                 0E   15  0176               BLEQ    155$                        ;BRANCH IF EMPTY
          50     04   C6  0178               DIVL    #4,R0                       ;FORM LONG WORD COUNT OF MAX TO DUMP
          58     50   D1  017B               CMPL    R0,R8                       ;USE SMALLER OF MAX AND DEFAULT
                 03   18  017E               BGEQ    145$                        ;
          58     50   D0  0180               MOVL    R0,R8                       ;USE THE MAX
               0182   30  0183      145$:    BSBW    DUMP_ARRAY                  ;DUMP KERNEL, INTERRUPT, OR EXEC STACK
       5B 5C     1F   E0  0186      155$:    BBS     #31,AP,190$                 ;DO NOT TRY FOR EXEC STACK IF SYSTEM SPACE
          5C     01   DB  018A               MFPR    #PR$_ESP,AP                 ;FETCH EXEC STACK POINTER
       A2 5A     16   E4  018D               BBSC    #PSL$V_PRVMOD,R10,126$      ;IF HAVEN'T DUMPED EXEC STACK, DO IT NOW
51     0000001A'EF   9E  0191               MOVAB   PRCPRV_MSG,R1               ;"PROCESS PRIVILEGES"
              FE65'   30  0198               BSBW    EXE$OUTCSTRING              ;OUTPUT COUNTED STRING
5C     00000000'9F   D0  019B               MOVL    @#SCH$GL_CURPCB,AP          ;CURRENT PROCESS CONTROL BLOCK ADDRESS
     5C      6C AC   D0  01A2               MOVL    PCB$L_PHD(AP),AP            ;PROCESS HEADER ADDRESS
                 06   18  01A6               BGEQ    170$                        ;IF NOT NEGATIVE, DON'T TRY TO USE IT
          58     02   D0  01A8               MOVL    #2,R8                       ;2 LONG WORDS AT FRONT OF HEADER
               015A   30  01AB               BSBW    DUMP_ARRAY                  ;OUTPUT THE PROCESS PRIVILEGES
51     00000037'EF   9E  01AE      170$:    MOVAB   IMGNAM_MSG,R1               ;"IMAGE NAME = "
              FE48'   30  01B5               BSBW    EXE$OUTCSTRING              ;OUTPUT THE COUNTED STRING
5C     00000000'9F   9E  01B8               MOVAB   @#CTL$GL_IMGHDRBF,AP        ;GET POINTER TO IMAGE HEADER BUFFER
                 58   D4  01BF               CLRL    R8                          ;DO NOT DUMP ANY DATA
               0144   30  01C1               BSBW    DUMP_ARRAY                  ;JUST CHECK FOR ACCESSABILITY
             1B  50   E9  01C4               BLBC    R0,180$                     ;BRANCH IF CANNOT ACCESS THE POINTER
          5C     6C   D0  01C7               MOVL    (AP),AP                     ;GET IMAGE HEADER BUFFER ADDRESS
                 16   13  01CA               BEQL    180$                        ;IF EQL, NO IMAGE CURRENTLY ACTIVE
                 58   D4  01CC               CLRL    R8                          ;DO NOT DUMP ANY DATA
               0137   30  01CE               BSBW    DUMP_ARRAY                  ;JUST CHECK FOR ACCESSABILITY
             0E  50   E9  01D1               BLBC    R0,180$                     ;BRANCH IF CANNOT ACCESS THE IMAGE HDR BUF
     51      04 AC   D0  01D4               MOVL    4(AP),R1                    ;ADDRESS OF IMAGE FILE DESCRIPTOR
     50      02 A1   3C  01D8               MOVZWL  IFD$W_FILNAMOFF(R1),R0      ;OFFSET TO NAME OF IMAGE BEING RUN
          51     50   C0  01DC               ADDL    R0,R1                       ;ADDRESS OF ASCIC NAME
              FE1E'   30  01DF               BSBW    EXE$OUTCSTRING              ;OUTPUT THE IMAGE NAME
              FE1B'   30  01E2      180$:    BSBW    EXE$OUTCRLF                 ;OUTPUT CARRIAGE RETURN, LINE FEED PAIR
                 8E   D5  01E5      190$:    TSTL    (SP)+                       ;REMOVE BUG CHECK CODE FROM STACK
                          01E7      ;
                          01E7      ; OUTPUT TO CONSOLE, IF ANY, IS FINISHED.  NOW WRITE OUT THE DUMP FILE.
                          01E7      ;
                          01E7      CONSOLE_DONE:
          7FFF 8F     BA  01E7               POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP,FP,SP> ;RESTORE
```

```
                00'  E0  01EB              BBS     S^#EXE$V_BUGREBOOT,-     ;CHECK FOR REBOOT
     06 00000000'9F      01ED                      @#EXE$GL_FLAGS,10$
        00000000'9F  16  01F3              JSB     @#INI$BRK                ;STOP IN XDELTA IF PRESENT
                         01F9      10$:                                     ;CHECK AND WRITE THE DUMP FILE
            7FFF 8F  BB  01F9              PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP,FP,SP> ;STACK REGS
                         01FD                                               ;FOR DISK WRITE
                00'  E1  01FD              BBC     S^#EXE$V_BUGDUMP,-       ; BRANCH IF NO DUMP
     11 00000000'9F      01FF                      @#EXE$GL_FLAGS,20$
                         0205      ;
                         0205      ; THE BOOT CONTROL BLOCK HAS ALREADY BEEN VALIDATED, JUST CHECK THAT
                         0205      ; A DUMP FILE IS ACTUALLY PRESENT.
                         0205      ;
   5A   00000000'9F  D0  0205              MOVL    @#EXE$GL_BOOTCB,R10      ;BOOT CONTROL BLOCK ADDRESS
          55   20 AA D0  020C              MOVL    BOO$L_DMP_MAP(R10),R5    ;VIRTUAL TO LOGICAL MAP FOR DUMP FILE
          59   1C AA D0  0210              MOVL    BOO$L_DMP_SIZE(R10),R9   ;SIZE OF DUMP FILE IN BLOCKS
                  03 14  0214              BGTR    30$                      ;BRANCH IF SOME BLOCKS ARE PRESENT
                00AE 31  0216      20$:    BRW     NODUMP                   ;NO DUMP
   56   00000000'9F  D0  0219      30$:    MOVL    @#EXE$GL_RPB,R6          ;GET BASE OF RESTART PARAMETER BLOCK
          55   34 A6 D0  0220              MOVL    RPB$L_IOVEC(R6),R5       ;FETCH POINTER TO BOOTDRIVER
        53    FBD8 CF 9E 0224              MOVAB   FATALBUG-512,R3          ;GET ADDRESS OF DUMP HEADER BLK BUFFER
                57 53 D0 0229              MOVL    R3,R7                    ;SET BUFFER ADDRESS FOR WRITEDUMP
                         022C              ASSUME  DMP$L_ERRSEQ EQ 0
   83   00000000'9F  D0  022C              MOVL    @#ERL$GL_SEQUENCE,(R3)+  ;SAVE ERROR LOG SEQUENCE NUMBER
                         0233              ASSUME  DMP$W_FLAGS EQ DMP$L_ERRSEQ+4
                  83 B4  0233              CLRW    (R3)+                    ;SET DUMP FILE FLAGS
                         0235              ASSUME  DMP$W_DUMPVER EQ DMP$W_FLAGS+2
             83   02 B0  0235              MOVW    #2,(R3)+                 ;SET DUMP FILE VERSION NUMBER
                         0238              ASSUME  DMP$L_SBR EQ DMP$W_DUMPVER+2
             83   0C DB  0238              MFPR    #PR$_SBR,(R3)+           ;SET SYSTEM BASE REGISTER
                         023B              ASSUME  DMP$L_SLR EQ DMP$L_SBR+4
             83   0D DB  023B              MFPR    #PR$_SLR,(R3)+           ;SET SYSTEM LENGTH REGISTER
                         023E              ASSUME  DMP$L_KSP EQ DMP$L_SLR+4
             83   00 DB  023E              MFPR    #PR$_KSP,(R3)+           ;SET KERNEL STACK POINTER
                         0241              ASSUME  DMP$L_ESP EQ DMP$L_KSP+4
             83   01 DB  0241              MFPR    #PR$_ESP,(R3)+           ;SET EXEC STACK POINTER
                         0244              ASSUME  DMP$L_SSP EQ DMP$L_ESP+4
             83   02 DB  0244              MFPR    #PR$_SSP,(R3)+           ;SET SUPER STACK POINTER
                         0247              ASSUME  DMP$L_USP EQ DMP$L_SSP+4
             83   03 DB  0247              MFPR    #PR$_USP,(R3)+           ;SET USER STACK POINTER
                         024A              ASSUME  DMP$L_ISP EQ DMP$L_USP+4
             83   04 DB  024A              MFPR    #PR$_ISP,(R3)+           ;SET INTERRUPT STACK POINTER
                         024D      ;
                         024D      ; IF THE RPB WAS CREATED BY A VERSION OF VMB LESS THAN 3, THEN
                         024D      ; CREATE A DUMMY MEMORY DESCRIPTOR FOR MAIN MEMORY BY ASSUMING
                         024D      ; THAT THE SPT RESIDES AT THE END OF PHYSICAL MEMORY
                         024D      ;
          50   10 A5 B2  024D              MCOMW   BOO$W_VERSION(R5),R0     ;GET VMB VERSION NUMBER 1'S COMPLEMENTED
          12 A5   50 B1  0251              CMPW    R0,BOO$W_VERSION+2(R5)   ;CHECK AGAINST CHECK WORD IN VMB
                  06 12  0255              BNEQ    40$                      ;IF NOT, ASSUME NO VERSION NUMBER
          03   10 A5 B1  0257              CMPW    BOO$W_VERSION(R5),#3     ;VERSION 3 OF VMB?
                  15 1E  025B              BGEQU   50$                      ;IF OK, USE DESCRIPTORS IN RPB
             50   0D DB  025D      40$:    MFPR    #PR$_SLR,R0              ;GET LENGTH OF SPT IN LONGWORDS
             51   0C DB  0260              MFPR    #PR$_SBR,R1              ;GET PHYSICAL ADDRESS OF SPT
           50   6140 DE  0263              MOVAL   (R1)[R0],R0              ;COMPUTE TOTAL PHYSICAL MEMORY SIZE
 00BC C6   50   F7 8F 78 0267              ASHL    #-9,R0,RPB$L_MEMDSC(R6)  ;STORE IN MEM. DESCRIPTOR PAGCNT,TR=0
                         026E              ASSUME  RPB$V_PAGCNT EQ 0
                         026E              ASSUME  RPB$V_TR EQ <RPB$V_PAGCNT+24>
```

```
                              026E            ASSUME  RPB$C_MEMDSCSIZ EQ 8
          00C0 C6       7C    026E            CLRQ    RPB$L_MEMDSC+4(R6)        ;SET STARTPFN=0 AND STORE 0 TERMINATOR
                              0272            ASSUME  RPB$V_BASEPFN EQ 32
                              0272            ASSUME  RPB$C_NMEMDSC GE 2
                              0272    ;
                              0272    ; COPY THE MEMORY DESCRIPTORS FROM THE RPB INITIALIZED IN VMB
                              0272    ;
                              0272            ASSUME  RPB$C_NMEMDSC EQ DMP$C_NMEMDSC
                              0272            ASSUME  RPB$C_MEMDSCSIZ EQ DMP$C_MEMDSCSIZ
   63    00BC C6    0040 8F    28    0272    50$:    MOVC3   #<RPB$C_NMEMDSC*RPB$C_MEMDSCSIZ>,RPB$L_MEMDSC(R6),(R3) ;SET THE
                              027A                                              ;MEMORY DESCRIPTORS FROM THE RPB
                              027A    ;
                              027A    ; STORE THE SYSTEM VERSION AND ONE'S COMPLEMENT CHECKSUM IN HEADER
                              027A    ;
      63     00000000'8F    D0    027A            MOVL    #SYS$K_VERSION,(R3)       ;SET THE VERSION # OF THE SYSTEM
         04 A3    63        D2    0281            MCOML   (R3),4(R3)                ;SET CHECK FIELD = ONES COMPLEMENT
                              0285    ;
                              0285    ; WRITE THE FIRST BLOCK OF THE DUMP FILE (THE HEADER)
                              0285    ;
      58    017C 8F     3C    0285            MOVZWL  #<EMB$K_CR_LENGTH+EMB$K_LENGTH+DMP$C_LENGTH>,R8 ;BUFFER SIZE
        55    20 AA     D0    028A            MOVL    BOO$L_DMP_MAP(R10),R5     ;VIRTUAL TO LOGICAL MAP FOR DUMP FILE
        5A    18 AA     D0    028E            MOVL    BOO$L_DMP_VBN(R10),R10    ;STARTING VBN OF DUMP FILE
             00E6       30    0292            BSBW    WRITEDUMP                 ;WRITE DUMP HEADER
                              0295    ;
                              0295    ; WRITE THE NEXT 2 BLOCKS OF ERROR LOG BUFFERS
                              0295    ;
      58    0400 8F     3C    0295            MOVZWL  #<2*512>,R8               ;SET SIZE FOR ERROR LOG BUFFERS
   57     00000000'9F   D0    029A            MOVL    @#ERL$AL_BUFADDR,R7       ;AND BUFFER ADDRESS
             00D7       30    02A1            BSBW    WRITEDUMP                 ;WRITE ERROR LOG BUFFERS
                              02A4    ;
                              02A4    ; NOW WRITE EVERY PAGE OF EVERY MEMORY OUT TO THE DUMP FILE.  VMB HAS BUILT
                              02A4    ; MEMORY DESCRIPTORS INTO THE RPB.  EACH DESCRIPTOR GIVES THE TR NUMBER, BASE
                              02A4    ; PFN, AND PAGE COUNT FOR A PARTICULAR MEMORY.  THERE MAY BE UP TO EIGHT MEMORY
                              02A4    ; DESCRIPTORS.  A DESCRIPTOR WITH A ZERO PAGE COUNT AND TR NUMBER INDICATES NO
                              02A4    ; MORE DESCRIPTORS.
                              02A4    ;
                              02A4            ASSUME  RPB$C_MEMDSCSIZ EQ 8
          54      08    D0    02A4            MOVL    #RPB$C_NMEMDSC,R4         ;GET MAXIMUM # OF MEM DESC POSSIBLE
     5B   00BC C6       9E    02A7            MOVAB   RPB$L_MEMDSC(R6),R11      ;GET ADR OF FIRST MEM DESC
58   6B    18     00    EF    02AC    60$:    EXTZV   #RPB$V_PAGCNT,#RPB$S_PAGCNT,(R11),R8 ;GET PAGCNT FOR THIS MEM
                  14    13    02B1            BEQL    NODUMP                    ;BR IF MEM DSC NOT USED
     58    58     09    78    02B3            ASHL    #9,R8,R8                  ;CONVERT PAGE COUNT TO BYTE COUNT
                              02B7            ASSUME  <RPB$S_PAGCNT + RPB$S_TR> EQ 32
          5B      04    C0    02B7            ADDL    #4,R11                    ;POINT TO BASE PFN IN MEMORY DESC
                              02BA            ASSUME  RPB$S_BASEPFN EQ 32
                              02BA            ASSUME  RPB$V_BASEPFN EQ 32
          57      8B    D0    02BA            MOVL    (R11)+,R7                 ;GET BASE PFN FOR THIS MEMORY
     57    57     09    78    02BD            ASHL    #9,R7,R7                  ;CONVERT PFN TO PHYSICAL ADDRESS
                00B7    30    02C1            BSBW    WRITEDUMP                 ;DUMP PAGES FOR THIS MEMORY
               E5 54    F5    02C4            SOBGTR  R4,60$                    ;LOOK FOR ANOTHER MEMORY DESCRIPTOR
                              02C7    ;
                              02C7    ; CHECK THE VMB VERSION NUMBER.  IF IT EXISTS AND IF IT IS 10 OR GREATER,
                              02C7    ; THEN CALL A UNIT DISCONNECT ROUTINE TO DO ANY DEVICE/UNIT SPECIFIC
                              02C7    ; DISCONNECT/DISMOUNT FOR THE SYSTEM DEVICE.
                              02C7    ;
                              02C7    NODUMP:
   59     00000000'9F   D0    02C7            MOVL    @#EXE$GL_RPB,R9           ;PICK UP THE RPB POINTER
```

```
          51    34 A9    D0  02CE       MOVL    RPB$L_IOVEC(R9),R1          ;FETCH POINTER TO BOOTDRIVER
          50    10 A1    B2  02D2       MCOMW   BQO$W_VERSION(R1),R0        ;GET VMB VERSION NUMBER 1'S COMPLEMENTED
          12 A1    50    B1  02D6       CMPW    R0,BQO$W_VERSION+2(R1)      ;CHECK AGAINST CHECK WORD IN VMB
                   10    12  02DA       BNEQ    10$                         ;IF NOT, ASSUME NO VERSION NUMBER
          0A    10 A1    B1  02DC       CMPW    BQO$W_VERSION(R1),#10       ;VERSION 10 OR GREATER OF VMB?
                   0A    1F  02E0       BLSSU   10$                         ;NO, DON'T CALL THE NON-EXISTENT CODE
          55    2C A1    D0  02E2       MOVL    BQO$L_UNIT_DISC(R1),R5      ;YES, IS THE ROUTINE PRESENT?
                   04    13  02E6       BEQL    10$                         ;NO, DON'T CALL
           6145    00    FB  02E8       CALLS   #0,(R1)[R5]                 ;DO IT!
                             02EC  ;
                             02EC  ; DONE EVERYTHING, NOW REBOOT THE SYSTEM OR SHUT IT DOWN
                             02EC  ;
 06 00000000'9F    00'   E1  02EC  10$:  BBC     S^#EXE$V_BUGREBOOT,@#EXE$GL_FLAGS,20$   ;BRANCH IF NO REBOOT
        0000015E'9F      17  02F4       JMP     @#REBOOT                    ;REBOOT THE PROCESSOR
                   5B    D4  02FA  20$:  CLRL    R11                         ;SET FOR CONSOLE TERMINAL OUTPUT
    51  0000004C'EF      9E  02FC       MOVAB   SHUT_DOWN,R1                ;SET ADDRESS OF MESSAGE STRING
                 FCFA'   30  0303       BSBW    EXE$OUTZSTRING              ;AND OUTPUT IT TO THE CONSOLE
                   FE    11  0306  30$:  BRB     30$                         ;LOOP FOREVER
                             0308
```

```
                              0308             .SBTTL  DUMP_ARRAY - SUBROUTINE TO DUMP AN ARRAY OF MEMORY LOCATIONS
                              0308      ;
                              0308      ; DUMP AN ARRAY OF MEMORY LOCATIONS WITH THEIR ADDRESSES AND CONTENTS
                              0308      ;
                              0308      ; INPUTS:
                              0308      ;
                              0308      ;       R8 = NUMBER OF LONG WORDS TO DUMP
                              0308      ;            IF 0 IS SPECIFIED THE FIRST ADDRESS IS CHECKED FOR RESIDENCY
                              0308      ;       AP = ADDRESS OF FIRST LONG WORD TO DUMP
                              0308      ;
                              0308      ; OUTPUTS:
                              0308      ;
                              0308      ;       R0 = LOW BIT SET IF SUCCESSFUL
                              0308      ;          = LOW BIT CLEAR IF CANNOT ACCESS THE ADDRESS IN AP
                              0308      ;       AP = ADDRESS OF NEXT LONG WORD NOT DUMPED
                              0308      ;       R4,R5,R8 ALTERED
                              0308      ;
                              0308      DUMP_ARRAY:
   54   00000000'9F      9E   0308              MOVAB   @#MMG$AL_SYSPCB,R4          ;PCB ADDRESS IF SYSTEM SPACE
          07 5C   1F     E0   030F              BBS     #31,AP,20$                  ;BRANCH IF SYSTEM SPACE
   54   00000000'9F      D0   0313              MOVL    @#SCH$GL_CURPCB,R4          ;PROCESS PCB FOR PROCESS SPAC
        55   6C A4       D0   031A      20$:    MOVL    PCB$L_PHD(R4),R5            ;CORRESPONDING PROCESS HEADER ADDRESS
                  1B     11   031E              BRB     70$                         ;LOOP 0 OR MORE TIMES
           50     09     9A   0320      60$:    MOVZBL  #^A/    /,R0                ;GET TAB CHARACTER
              FCDA'      30   0323              BSBW    EXE$OUTCHAR                 ;OUTPUT TAB CHARACTER
           51     5C     D0   0326              MOVL    AP,R1                       ;GET ADDRESS OF LONGWORD TO CONVERT
              FCD4'      30   0329              BSBW    EXE$OUTHEX                  ;CONVERT ADDRESS OF LONGWORD
              FCD1'      30   032C              BSBW    EXE$OUTBLANK                ;OUTPUT BLANK CHARACTER
              FCCE'      30   032F              BSBW    EXE$OUTBLANK                ;OUTPUT BLANK CHARACTER
           51     8C     D0   0332              MOVL    (AP)+,R1                    ;GET CONTENTS OF LONGWORD TO OUTPUT
              FCC8'      30   0335              BSBW    EXE$OUTHEX                  ;OUTPUT CONVERTED HEX LONGWORD
              FCC5'      30   0338              BSBW    EXE$OUTCRLF                 ;OUTPUT CARRIAGE RETURN, LINE FEED PAIR
           52     5C     D0   033B      70$:    MOVL    AP,R2                       ;MAKE SURE THAT THIS ADDRESS CAN BE ACCESSED
        00000000'9F      16   033E              JSB     @#MMG$PTEINDX               ;GET LONG WORD INDEX TO SVAPTE IN R3
              31 50      E9   0344              BLBC    R0,100$                     ;BRANCH IF LENGTH VIOLATION
        53   6543        DE   0347              MOVAL   (R5)[R3],R3                 ;FORM SYSTEM VIRTUAL ADR OF PTE
                  63     D5   034B              TSTL    (R3)                        ;SEE IF PAGE TABLE ENTRY IS VALID
                  22     19   034D              BLSS    75$                         ;BRANCH IF IT IS
     02 A3    0440 8F    B3   034F              BITW    #<PTE$M_TYP1 ! PTE$M_TYP0>@-16,2(R3) ;IF TRANSITION PAGE
                  21     12   0355              BNEQ    100$                        ;BRANCH IF NOT
 50     63    15  00     EF   0357              EXTZV   #PTE$V_PFN,#PTE$S_PFN,(R3),R0 ;GET PAGE FRAME NUMBER
                  1A     13   035C              BEQL    100$                        ;BRANCH IF DEMAND ZERO FORMAT
        00000000'9F      DD   035E              PUSHL   @#PFN$AB_STATE              ;ADDRESS OF STATE TABLE
             03   00     ED   0364              CMPZV   #PFN$V_LOC,#PFN$S_LOC,-     ;PAGE IS OK IN MEMORY UNLESS
          06    9E40          0367                      @(SP)+[R0],#PFN$C_RDINPROG ;IT IS BEING READ IN
                  0C     13   036A              BEQL    100$
      03 A3    80 8F     88   036C              BISB    #<PTE$M_VALID>@-24,3(R3) ;JAM IT VALID AND USE IT
              AC 58      F4   0371      75$:    SOBGEQ  R8,60$                      ;ANY MORE LONGWORDS TO CONVERT?
           50     01     D0   0374      80$:    MOVL    #1,R0                       ;INDICATE SUCCESSFUL COMPLETION
                         05   0377              RSB
                              0378      ;
                              0378      ; CANNOT ACCESS ADDRESS POINTED TO BY AP
                              0378      ;
                  50     D4   0378      100$:   CLRL    R0
                         05   037A              RSB
```

```
                          037B          .SBTTL  WRITEDUMP - WRITE DATA TO DUMP FILE
                          037B   ;
                          037B   ; WRITE DATA TO SYSTEM DUMP FILE
                          037B   ;
                          037B   ; INPUTS:
                          037B   ;              R5 - ADDRESS OF VIRTUAL TO LOGICAL MAP FOR DUMP FILE
                          037B   ;              R6 - ADDRESS OF RESTART PARAMETER BLOCK
                          037B   ;              R7 - BUFFER ADDRESS
                          037B   ;              R8 - SIZE OF BUFFER IN BYTES
                          037B   ;              R9 - NUMBER OF BLOCKS NOT YET WRITTEN IN DUMP FILE
                          037B   ;              R10 - VBN OF DUMP FILE (UPDATED)
                          037B   ;
                          037B   ; OUTPUTS:
                          037B   ;              R7 - UPDATED
                          037B   ;              R8 - UPDATED
                          037B   ;              R9 - UPDATED
                          037B   ;              R10 - UPDATED
                          037B   ;
           0000FE00       037B          IOSIZE=127*512                           ;MAXIMUM TRANSFER SIZE
                          037B   WRITEDUMP:                                      ;
           0420 8F    BB  037B          PUSHR   #^M<R5,R10>                      ;SAVE MAP AND VBN
 52    85    FD 8F    78  037F          ASHL    #-3,(R5)+,R2                     ;COUNT OF RETRIEVAL POINTERS
       50    85       7D  0384   10$:   MOVQ    (R5)+,R0                         ;R0=BLOCK COUNT, R1=STARTING LBN
       50    5A       D1  0387          CMPL    R10,R0                           ;VBN COVERED BY THIS RETRIEVAL POINTER?
             0C       15  038A          BLEQ    20$                              ;BRANCH IF YES
       5A    50       C2  038C          SUBL    R0,R10                           ;NO, REDUCE VBN BY BLOCKS PASSED OVER
             F2 52    F5  038F          SOBGTR  R2,10$                           ;TRY NEXT RETRIEVAL POINTER
           0420 8F    BA  0392          POPR    #^M<R5,R10>                      ;RESTORE MAP AND VBN
             6A       11  0396          BRB     100$                             ;EOF, NO MORE WRITING
             5A       D7  0398   20$:   DECL    R10                              ;MAKE VBN 0 ORIGIN
       50    5A       C2  039A          SUBL    R10,R0                           ;NO. OF BLOCKS AFTER DESIRED VBN
       51    5A       C0  039D          ADDL    R10,R1                           ;STARTING LBN OF DESIRED VBN
           0420 8F    BA  03A0          POPR    #^M<R5,R10>                      ;RESTORE MAP AND VBN
                          03A4   ;
                          03A4   ; R0 = NUMBER OF BLOCKS THAT COULD BE TRANSFERRED
                          03A4   ; R1 = STARTING LBN OF THE TRANSFER
                          03A4   ;
       53    FE00 8F  3C  03A4          MOVZWL  #IOSIZE,R3                       ;ASSUME MAXIMUM
       50    50    09 78  03A9          ASHL    #9,R0,R0                         ;BYTE COUNT THAT COULD BE TRANSFERRED
       50    53       D1  03AD          CMPL    R3,R0                            ;MINIMIZE WITH MAX LEGAL XFER
             03       15  03B0          BLEQ    30$
       53    50       D0  03B2          MOVL    R0,R3
       58    53       D1  03B5   30$:   CMPL    R3,R8                            ;MINIMIZE WITH BYTE COUNT
             03       15  03B8          BLEQ    40$                              ;REMAINING TO BE TRANSFERRED
       53    58       D0  03BA          MOVL    R8,R3
       52    01FF C3  9E  03BD   40$:   MOVAB   511(R3),R2                       ;ROUND UP BYTE COUNT AND FORM
 52    52    F7 8F    78  03C2          ASHL    #-9,R2,R2                        ;PAGES TO BE WRITTEN
             39       13  03C7          BEQL    100$                             ;NOTE NOTHING TO TRANSFER
       59    52       D1  03C9          CMPL    R2,R9                            ;MINIMIZE WITH PAGES LEFT IN FILE
             09       15  03CC          BLEQ    50$
       53    59    09 78  03CE          ASHL    #9,R9,R3                         ;USE BYTE COUNT REMAINING IN FILE
       52    59       D0  03D2          MOVL    R9,R2                            ;AND BLOCK COUNT TO TRANSFER
             2B       13  03D5          BEQL    100$                             ;BRANCH IF NO BLOCK LEFT IN FILE
             56       DD  03D7   50$:   PUSHL   R6                               ;SET ADDRESS OF RPB
 7E    57    01    1F EF  03D9          EXTZV   #VA$V_SYSTEM,#1,R7,-(SP);USE SYSTEM BIT AS VIRTUAL FLAG
             20       DD  03DE          PUSHL   S^#IO$_WRITELBLK                 ;SET FUNCTION CODE
             51       DD  03E0          PUSHL   R1                               ;LBN IN DUMP FILE
```

```
                  53  DD  03E2            PUSHL   R3                          ;SIZE OF BUFFER IN BYTES
                  57  DD  03E4            PUSHL   R7                          ;ADDRESS OF BUFFER
         50    34 A6  D0  03E6            MOVL    RPB$L_IOVEC(R6),R0          ;BOOT DRIVER VECTOR
         00 B040  06  FB  03EA            CALLS   #6,@BQO$L_QIO(R0)[R0]       ;CALL BOOTDRIVR
              57  53  C0  03EF            ADDL    R3,R7                       ;UPDATE BUFFER ADDRESS
              5A  52  C0  03F2            ADDL    R2,R10                      ;UPDATE VBN
              59  52  C2  03F5            SUBL    R2,R9                       ;AND SIZE OF FILE
                  08  15  03F8            BLEQ    100$                        ;DONE IF END OF FILE
              58  53  C2  03FA            SUBL    R3,R8                       ;UPDATE BYTE COUNT
                  03  15  03FD            BLEQ    100$                        ;DONE IF BYTE COUNT EXHAUSTED
                FF79  31  03FF            BRW     WRITEDUMP                   ;OTHERWISE START FROM THE TOP
                      05  0402   100$:    RSB                                 ;
```

```
                              0403          .SBTTL  SUBROUTINES TO BUILD HEADERS AND VERIFY BOOT CONTROL BLOCK
                              0403
                              0403  ;
                              0403  ; SUBROUTINE TO BUILD HEADER AND DUMP GENERAL REGISTERS
                              0403  ;
                              0403
                          00000178          .PSECT  $AEXENONPAGED
                              0178  BUILD_HEADER:                                   ;
                              0178          ASSUME  EMB$W_BC_ENTRY EQ EMB$W_CR_ENTRY
         04 A2     59     B0  0178          MOVW    R9,EMB$W_BC_ENTRY(R2)           ;SET TYPE OF ENTRY IN EMB
                              017C          ASSUME  EMB$L_BC_KSP EQ EMB$L_CR_KSP
         50    10 A2      9E  017C          MOVAB   EMB$L_BC_KSP(R2),R0             ;POINT TO PLACE IN EMB FOR 1ST REGISTER
                   0A     10  0180          BSBB    DUMP_REGISTERS                  ;INSERT PROCESSOR STACK POINTERS
                              0182          ASSUME  EMB$L_BC_R0 EQ EMB$L_BC_R0
                              0182          ASSUME  EMB$L_BC_PSL EQ EMB$L_CR_PSL
         51        11     D0  0182          MOVL    #<EMB$L_BC_PSL+4-EMB$L_BC_R0>/4,R1 ;SET NUMBER OF REGISTERS
         80        8C     D0  0185  10$:    MOVL    (AP)+,(R0)+                     ;INSERT GENERAL REGISTER
            FA     51     F5  0188          SOBGTR  R1,10$                          ;ANY MORE REGISTERS TO INSERT?
                   05         018B          RSB                                     ;
                              018C
                              018C  ;
                              018C  ; SUBROUTINE TO DUMP PROCESSOR REGISTERS UNTIL ESCAPE
                              018C  ;
                              018C
                              018C  DUMP_REGISTERS:                                 ;
         51        8B     98  018C          CVTBL   (R11)+,R1                       ;GET NEXT INTERNAL REGISTER NUMBER
                   05     19  018F          BLSS    RETURN                          ;IF LSS ESCAPE
         80        51     DB  0191          MFPR    R1,(R0)+                        ;INSERT PROCESSOR REGISTER
                   F6     11  0194          BRB     DUMP_REGISTERS                  ;
                   05         0196  RETURN: RSB                                     ;
                              0197
                              0197  ; VALIDATE THE CHECKSUM FOR THE BOOT CONTROL BLOCK
                              0197  ;  AND SYS.EXE WINDOW CONTROL BLOCK
                              0197  ;
                              0197  ; INPUTS:
                              0197  ;
                              0197  ;       EXE$GQ_BOOTCB_D a descriptor for the boot control block and SYS.EXE
                              0197  ;                       window control block.  The descriptor is assumed to
                              0197  ;                       delineate an area which includes both the boot control
                              0197  ;                       block and the SYS.EXE window control block.  The
                              0197  ;                       address field of the descriptor is assumed to point to
                              0197  ;                       the boot control block.
                              0197  ;
                              0197  ;       IPL >= IPL$_SYNCH (If IPL is lower than IPL$_SYNCH, the checksum
                              0197  ;                         calculation may be wrong.)
                              0197  ;
                              0197  ; OUTPUTS:
                              0197  ;
                              0197  ;       Z SET IF CHECKSUM MATCHES, Z CLEAR IF NOT
                              0197  ;       R1 = ADDRESS OF BEGINNING OF BOOT CONTROL BLOCK
                              0197  ;       R3 = DESIRED CHECKSUM VALUE
                              0197  ;       R0 ALTERED
                              0197  ;       ALL OTHER REGISTERS PRESERVED
                              0197  ;
                              0197
                              0197  EXE$BOOTCB_CHK::
  50    00000000'EF       7D  0197          MOVQ    EXE$GQ_BOOTCB_D,R0              ;GET DESCRIPTOR OF BLOCK TO CHECKSUM
```

```
   51    50  C0  019E             ADDL    R0,R1                     ;POINT TO END OF BOOT CONTROL BLOCK
   50    04  C6  01A1             DIVL    #4,R0                     ;FORM LONG WORD COUNT
                 01A4
                 01A4             ASSUME  BOO$L_CHECKSUM EQ 0
         50  D7  01A4             DECL    R0                        ;DON'T ADD FIRST LONG WORD
         53  D4  01A6             CLRL    R3                        ;INIT CHECKSUM
   53    71  C0  01A8     10$:    ADDL    -(R1),R3                  ;FORM ADDITIVE CHECKSUM
      FA 50  F5  01AB             SOBGTR  R0,10$                    ;LOOP THROUGH THE BLOCK
 50   10 A1  D0  01AE             MOVL    BOO$L_SYS_MAP-4(R1),R0    ; Get pointer to system WCB.
 53   24 A0  C2  01B2             SUBL    WCB$L_READS(R0),R3        ; Remove to varying WCB entries from
 53   28 A0  C2  01B6             SUBL    WCB$L_WRITES(R0),R3       ; the checksum.
   71    53  D1  01BA             CMPL    R3,-(R1)                  ;DOES THE CHECKSUM MATCH
         05      01BD             RSB
                 01BE
                 01BE             .END
```

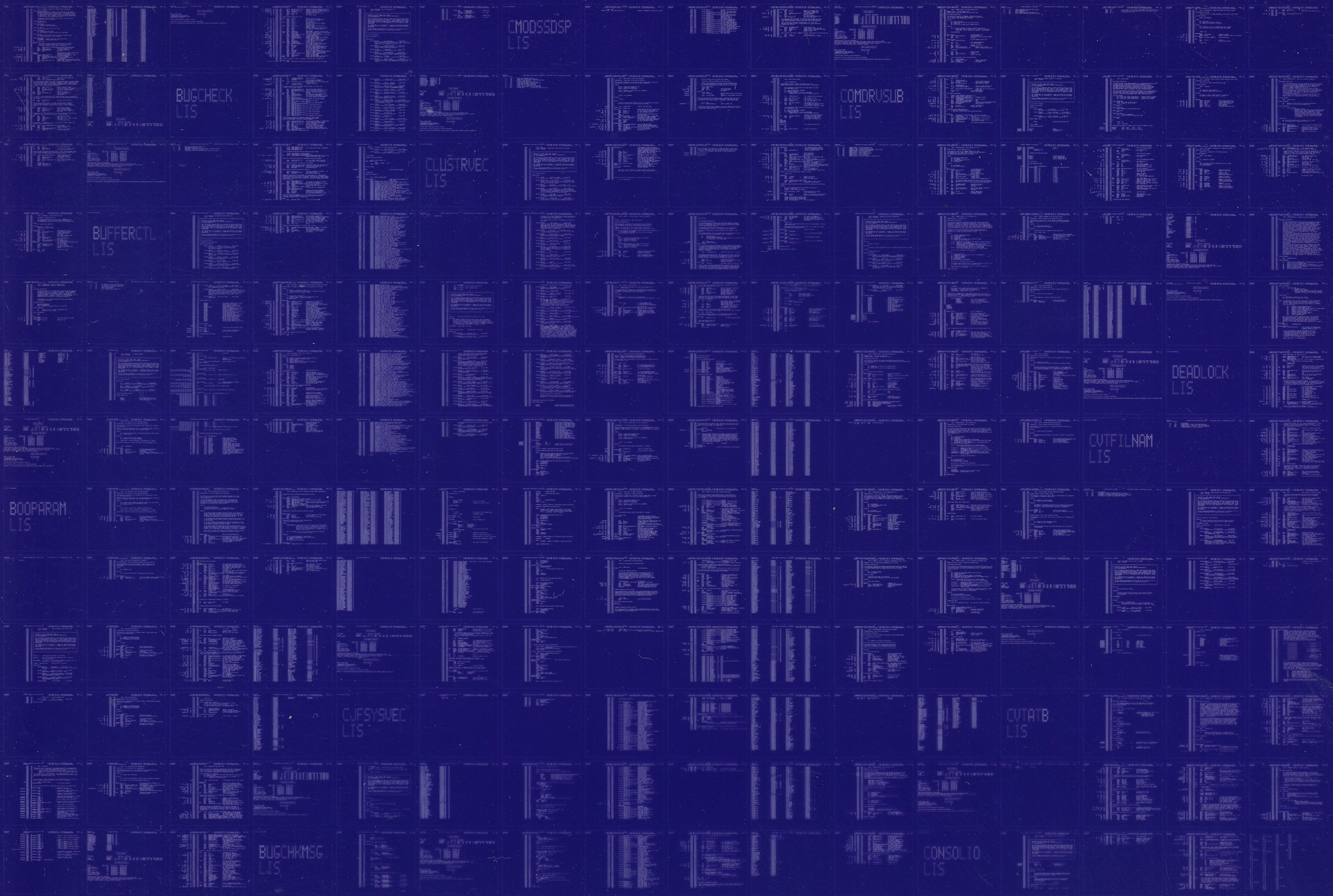
```
                                                  K  4
BUGCHECK                - SOFTWARE BUG CHECK ERROR LOGIC      16-SEP-1984 02:37:19  VAX/VMS Macro V04-00        Page  20
Symbol table                                                   5-SEP-1984 03:40:15  [SYS.SRC]BUGCHECK.MAR;1          (1)

BOO$L_BUG_MAP            = 00000024                EMB$Q_CR_TIME            = 00000006
BOO$L_CHECKSUM           = 00000000                EMB$T_BC_LNAME           = 00000070
BOO$L_DMP_MAP            = 00000020                EMB$T_CR_LNAME           = 000000FC
BOO$L_DMP_SIZE           = 0000001C                EMB$W_BC_ENTRY           = 00000004
BOO$L_DMP_VBN            = 00000018                EMB$W_CR_ENTRY           = 00000004
BOO$L_SYS_MAP            = 00000014                EMB$W_CR_ERRSEQ          = 0000000E
BQO$L_QIO                = 00000000                EMB$W_SIZE               = FFFFFFFC
BQO$L_UNIT_DISC          = 0000002C                ERL$ALLOCEMB               ********    X   06
BQO$L_UNIT_INIT          = 0000001C                ERL$AL_BUFADDR             ********    X   07
BQO$W_VERSION            = 00000010                ERL$GL_SEQUENCE            ********    X   07
BUG$A_PAGED                00000000 RG     07      ERL$RELEASEMB              ********    X   06
BUG$A_PAGEDEND             00000000 RG     04      EXE$ACVIOLAT               ********    X   06
BUG$FATAL                  00000000 RG     03      EXE$AL_STACKS              ********    X   07
BUG$T_MESSAGES             ********    X   07      EXE$BOOTCB_CHK             00000197 RG     06
BUG$_OPERATOR              ********    X   07      EXE$BUG_CHECK              00000012 RG     06
BUGCHK_FLAGS               00000000 R      02      EXE$DUMPCPUREG             ********    X   07
BUILD_HEADER               00000178 R      06      EXE$GL_BOOTCB              ********    X   06
CON$C_BOOTCPU            = 00000002                EXE$GL_BUGCHECK            00000008 RG     02
CON$OWNCTY                 ********    X   07      EXE$GL_FLAGS               ********    X   06
CON$SENDCONSCMD            ********    X   06      EXE$GL_RPB                 ********    X   06
CONSOLE_DONE               000001E7 R      07      EXE$GQ_BOOTCB_D            ********    X   06
CR                       = 0000000D                EXE$GQ_SYSTIME             ********    X   07
CTL$AL_STACK               ********    X   07      EXE$INIBOOTADP             ********    X   06
CTL$AL_STACKLIM            ********    X   07      EXE$OUTBLANK               ********    X   07
CTL$GL_IMGHDRBF            ********    X   07      EXE$OUTCHAR                ********    X   07
DMP$C_LENGTH             = 0000006C                EXE$OUTCRLF                ********    X   07
DMP$C_MEMDSCSIZ          = 00000008                EXE$OUTCSTRING             ********    X   07
DMP$C_NMEMDSC            = 00000008                EXE$OUTHEX                 ********    X   07
DMP$L_ERRSEQ             = 00000000                EXE$OUTZSTRING             ********    X   07
DMP$L_ESP                = 00000014                EXE$SHUTDWNADP             ********    X   06
DMP$L_ISP                = 00000020                EXE$V_BUGDUMP              ********    X   07
DMP$L_KSP                = 00000010                EXE$V_BUGREBOOT            ********    X   06
DMP$L_SBR                = 00000008                EXE$V_FATAL_BUG            ********    X   06
DMP$L_SLR                = 0000000C                EXE$V_SYSWRTABL            ********    X   06
DMP$L_SSP                = 00000018                FATALBUG                   00000000 R      07
DMP$L_USP                = 0000001C                FATAL_SPSAV                00000004 R      02
DMP$W_DUMPVER            = 00000006                IFD$W_FILNAMOFF          = 00000002
DMP$W_FLAGS              = 00000004                IMGNAM_MSG                 00000037 R      05
DUMP_ARRAY                 00000308 R      07      INI$BRK                    ********    X   06
DUMP_REGISTERS             0000018C R      06      INI$WRITABLE               ********    X   06
EMB$B_VALID              = FFFFFFFF                IO$_READLBLK             = 00000021
EMB$K_BC_LENGTH          = 00000080                IO$_WRITELBLK            = 00000020
EMB$K_CR                 = 00000025                IOSIZE                   = 0000FE00
EMB$K_CR_LENGTH          = 0000010C                LF                       = 0000000A
EMB$K_LENGTH             = 00000004                MMG$AL_SYSPCB              ********    X   07
EMB$K_SBC                = 00000028                MMG$PTEINDX                ********    X   07
EMB$K_UBC                = 00000070                MPH$BUGCHKHK               000000F6 RG     06
EMB$L_BC_CODE            = 00000068                MSGCTRL                    00000087 R      05
EMB$L_BC_KSP             = 00000010                MSGCTRL1                   000000AB R      05
EMB$L_BC_PID             = 0000006C                NODUMP                     000002C7 R      07
EMB$L_BC_PSL             = 00000064                PCB$L_PHD                = 0000006C
EMB$L_BC_R0              = 00000024                PCB$L_PID                = 00000060
EMB$L_CR_CODE            = 000000F4                PCB$Q_PRIV               = 00000084
EMB$L_CR_KSP             = 00000010                PCB$T_LNAME              = 00000070
EMB$L_CR_PID             = 000000F8                PFN$AB_STATE               ********    X   07
EMB$L_CR_PSL             = 00000064                PFN$C_RDINPROG           = 00000006
EMB$L_HD_SID             = 00000000                PFN$S_LOC                = 00000003
```

```
PFN$V_LOC                       = 00000000                      WCB$L_WRITES                    = 00000028
PR$_ASTLVL                      = 00000013                      WRITEDUMP                         0000037B R       07
PR$_ESP                         = 00000001
PR$_ICCS                        = 00000018
PR$_IPL                         = 00000012
PR$_ISP                         = 00000004
PR$_KSP                         = 00000000
PR$_P0BR                        = 00000008
PR$_P0LR                        = 00000009
PR$_P1BR                        = 0000000A
PR$_P1LR                        = 0000000B
PR$_PCBB                        = 00000010
PR$_SBR                         = 0000000C
PR$_SCBB                        = 00000011
PR$_SID                         = 0000003E
PR$_SISR                        = 00000015
PR$_SLR                         = 0000000D
PR$_SSP                         = 00000002
PR$_USP                         = 00000003
PRCNAM_MSG                        00000000 R       05
PRCPRV_MSG                        0000001A R       05
PRV$V_BUGCHK                    = 00000017
PSL$C_EXEC                      = 00000001
PSL$S_PRVMOD                    = 00000002
PSL$V_PRVMOD                    = 00000016
PTE$M_TYP0                      = 00400000
PTE$M_TYP1                      = 04000000
PTE$M_VALID                     = 80000000
PTE$S_PFN                       = 00000015
PTE$V_PFN                       = 00000000
READ_ERR_RETRY                    00000158 R       06
REBOOT                            0000015E R       06
REGTAB                            00000000 R       06
RETURN                            00000196 R       06
RPB$C_MEMDSCSIZ                 = 00000008
RPB$C_NMEMDSC                   = 00000008
RPB$L_IOVEC                     = 00000034
RPB$L_MEMDSC                    = 000000BC
RPB$S_BASEPFN                   = 00000020
RPB$S_PAGCNT                    = 00000018
RPB$S_TR                        = 00000008
RPB$V_BASEPFN                   = 00000020
RPB$V_PAGCNT                    = 00000000
RPB$V_TR                        = 00000018
SCH$GL_CURPCB                     ********   X     06
SCS$SHUTDOWN                      ********   X     06
SHUT_DOWN                         0000004C R       05
SS$_BUGCHECK                    = 000002A4
STS$K_ERROR                     = 00000002
STS$K_SEVERE                    = 00000004
STS$S_SEVERITY                  = 00000003
STS$V_SEVERITY                  = 00000000
SYS$EXIT                          ********  GX     06
SYS$GQ_VERSION                    ********   X     07
SYS$K_VERSION                     ********   X     07
VA$V_SYSTEM                     = 0000001F
WCB$L_READS                     = 00000024
```

```
                                          +----------------+
                                          ! Psect synopsis !
                                          +----------------+

PSECT name                Allocation           PSECT No.  Attributes
----------                ----------           ---------  ----------
.  ABS  .                 00000000  (     0.)  00 (   0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                     00000000  (     0.)  01 (   1.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$$$025                    0000000C  (    12.)  02 (   2.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$ZBUGFATAL                00000000  (     0.)  03 (   3.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC WORD
Z$INIT__BUGZEND           00000000  (     0.)  04 (   4.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC WORD
Z$INIT__BUGC              00000169  (   361.)  05 (   5.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$AEXENONPAGED             000001BE  (   446.)  06 (   6.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
Z$INIT__BUGA              00000403  (  1027.)  07 (   7.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC PAGE

                                       +------------------------+
                                       ! Performance indicators !
                                       +------------------------+

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                   29    00:00:00.08    00:00:00.25
Command processing              105    00:00:00.55    00:00:01.09
Pass 1                          495    00:00:19.22    00:00:22.42
Symbol table sort                 0    00:00:03.09    00:00:03.31
Pass 2                          183    00:00:03.93    00:00:04.37
Symbol table output              21    00:00:00.18    00:00:00.18
Psect synopsis output             3    00:00:00.05    00:00:00.05
Cross-reference output            0    00:00:00.00    00:00:00.00
Assembler run totals            838    00:00:27.10    00:00:31.68

The working set limit was 1950 pages.
110864 bytes (217 pages) of virtual memory were used to buffer the intermediate code.
There were 110 pages of symbol table space allocated to hold 1960 non-local and 56 local symbols.
886 source lines were read in Pass 1, producing 29 object records in Pass 2.
37 pages of virtual memory were used to define 36 macros.

                                      +--------------------------+
                                      ! Macro library statistics !
                                      +--------------------------+

Macro library name                       Macros defined
------------------                       --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                 21
_$255$DUA28:[SYSLIB]STARLET.MLB;2              12
TOTALS (all libraries)                         33

2084 GETS were required to define 33 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:BUGCHECK/OBJ=OBJ$:BUGCHECK MSRC$:BUGCHECK/UPDATE=(ENH$:BUGCHECK)+EXECML$/LIB
```

0373 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



CMODSSDSP
LIS

BUGCHECK
LIS

COMDRVSUB
LIS

CLUSTRVEC
LIS

BUFFERCTL
LIS

DEADLOCK
LIS

CVTFILNAM
LIS

BOOPARAM
LIS

CJFSYSVEC
LIS

CVTATB
LIS

BUGCHKMSG
LIS

CONSOLIO
LIS